It is Time to propose a Complete Methodology for Designing Semantic Databases

Chedlia Chakroun[1], Ladjel Bellatreche[1] and Yamine Aït-Ameur[2]
[1]*LIAS - ISAE ENSMA and University of Poitiers, Futuroscope, France*
[2]*IRIT - ENSEEIHT, Toulouse, France*

Keywords: Ontology, Database, Semantic Database, Dependencies, Design.

Abstract: Nowadays, domain ontologies have been advocated by advanced applications. This phenomenon contributes to the generation of big semantic data hard to manage by the main memory solutions. Persistent solutions were proposed as an alternative to ensure the scalability of those applications. As a consequence, a new type of database is born, called semantic database ($\mathcal{SDB}$). Several types of $\mathcal{SDB}$ have been proposed including different architectures of the target DBMS and storage models for ontology and its instances. By exploring the literature, we figure out that most important research studies were concentrated on the physical design of the $\mathcal{SDB}$, where solutions were proposed to increase the performance of the final $\mathcal{SDB}$. In this paper, we propose a design methodology dedicated to $\mathcal{SDB}$ including the main phases of the lifecycle of the database development: conceptual, logical and physical. The conceptual design of $\mathcal{SDB}$ can be easily performed by exploiting the similarities between ontologies and conceptual models. The logical design phase is performed thanks to the incorporation of dependencies between concepts and properties of the ontologies. Finally, a prototype implementing our design approach on Oracle 11g is outlined.

1 INTRODUCTION

Recently, ontologies have been widely adopted by small, medium and large companies in various domains such as Semantic Web, Web services, engineering, etc. This is due to three main characteristics: (1) the capability of ontologies to explicit semantic of concepts and instances manipulated by applications consuming ontologies; (2) the similarities between conceptual models and ontologies and (3) the development and the diversity of ontologies models adapted to large scale of application domains. Based on these characteristics offered by ontologies and their models, we can say that the ontologies leverage the conceptual data models proposed by Peter Chen. Recall that he argued that the world may be modeled by the use of two concepts, named, entity and relationship. Contrary to conceptual models, ontologies brought two main issues: (1) An ontology may contain non-canonical concepts (concepts derived from other ones i.e., notions expressed in term of other concepts), whereas, a conceptual model stores only canonical concepts (the primitive concepts). (2) An ontology offers the reasoning capabilities. Based on this discussion, we can claim that if a domain ontology exists, the designer tasks may be reduced. This is because she/he can use it as a basis to generate her/his conceptual model, since properties and concepts are already materialized in that ontology.

On the other hand, by exploring the research efforts, we figured out that they were initially focused on how to store, manipulate and query ontologies and their instances. We can say that these efforts were mainly concentrated on the physical design phase of semantic applications. At first, semantic data were managed and manipulated in the main memory. These solutions suffered from the scalability problem. Afterward, persistent solutions were proposed which gave raise to a new type of databases, called semantic databases ($\mathcal{SDB}$). Academicians and industrials propose a large panoply of $\mathcal{SDB}$ (Broekstra et al., 2002; Das et al., 2004; Dehainsala et al., 2007; Pan and Heflin, 2003). Three main architectures of $\mathcal{SDB}$ were proposed. The $Type_1$ and $Type_2$ architectures hard-coded the ontology model (RDF or RDFS, etc.). while in the $Type_3$ architecture, a new part, called the *meta-schema part* was added (Dehainsala et al., 2007). The presence of the meta-schema offers a support of the used ontology model's evolution by adding non functional properties such as preferences,

DOI: 10.5220/0004372901600164

web services, etc. Other research efforts developed in the physical design phase were concentrated on proposing storage schemas for ontologies and their instances. Three main storage layouts are identified: vertical, horizontal and hybrid (Pan and Heflin, 2003). Note that the existing $\mathcal{SDB}$ use fixed storage models to store ontologies and their data. For example, Oracle (Das et al., 2004) adopts the vertical representation where ontologies and data are stored together in a single triple table. Therefore, the storage mechanism consists in a direct load of ontologies and their data in frozen storage models without checking the redundancy and/or the inconsistency of the data. As a consequence, current $\mathcal{SDB}$ suffer specially from the presence of duplicated and inconsistent data. The presence of these anomalies can be explained by the lack of effort given to the design stage. Existing $\mathcal{SDB}$ approaches focus mainly on reasoning. Yet, the efficient storage is not treated entirely as a main issue as in traditional databases where design methodologies have been proposed. In these proposed approaches, anomalies are generally managed thanks to the exploitation of the available functional dependencies (FD) defined on the attributes. In fact, FD provide an elegant formalism to specify key constraints and present the basis for normalization process. Unfortunately, the dependencies relationships are not handled in the $\mathcal{SDB}$ design process.

By examining the ontological concepts definitions, we have identified a set of deductible characteristics that are close to classical FD in relational databases. Among these ontological concepts, we are interested in those describing dependency relationships between any ontological concepts being either a class or an attribute. Two types of dependencies are identified: (1) dependency relationships between properties and (2) dependency relationships between classes. Since fixed and static logical database schemes have been proposed to store ontological data, we deduce that these dependency relationships have not been exploited for the design of efficient $\mathcal{SDB}$ models. Indeed, just like traditional database, the dependency relationships between ontological concepts can play a crucial role in the process of designing databases dedicated to the ontology persistence. They may have a significant impact on the phase of the logical modeling and normalization process and thereafter to regain control of the $\mathcal{SDB}$ design process. Ignoring the logical phase in designing $\mathcal{SDB}$ may generate inconsistent and/or duplicated data.

In this paper, we propose to describe our work which consists in:

- exploiting the fact that ontology carries non-canonical concepts to identify conceptual dependencies,
- exploiting them to propose a process for $\mathcal{SDB}$ design which takes into account the nature of the source ontology, rather than having a fixed and static logical schema,
- validating our approach on a particular $\mathcal{SDB}$.

The remainder of this paper is organized as follows. Section 2 presents the related works. In Section 3, we propose our approach exploiting conceptual dependencies to improve the semantic database process. Section 4 shows an application of our approach in a particular $\mathcal{SDB}$: Oracle 11g. Finally, section 5 gives a conclusion and some future research directions.

2 RELATED WORK

Since the 70s, the FD have been widely studied in the database theory. These dependencies, usually defined on the attributes, were especially exploited in the databases design process. They are used to model the relationships between attributes of a relation, compute primary keys, define the normalized logical model, check the data consistency, etc. For the description logics (DL), FD have also been the subject of several studies (Borgida and Weddell, 1997; Calvanese et al., 2008; Motik et al., 2009; Romero et al., 2009; Calbimonte et al., 2009). In (Borgida and Weddell, 1997), Borgida et al. have expressed the need to add unique constraints for semantic data models, particularly for the description logic, while in (Calvanese et al., 2008), the authors studied the possibility to make them explicit in this language. In (Motik et al., 2009), the authors showed the role of constraints in the ontologies while drawing a comparison between the constraints in databases and those in ontologies. In (Calbimonte et al., 2009; Romero et al., 2009), the authors were interested in the study of dependency relationships and their implications in ontologies. In (Calbimonte et al., 2009), the authors propose a new OWL constructor to define FD while in (Romero et al., 2009), the authors propose an approach to define FD for a domain ontology based on the concepts and roles defined in such ontology. Romero et al. introduce a functional dependency as a relationship between classes. For two concepts C_1 and C_2, the authors established that each instance $i_1 \in C_1$ determines a single instance of C_2 if (1) there exists a functional role (r_i) valued for i_1 and (2) r_i connects i_1 to a unique instance of C_2. This dependency relationship is denoted by $C_1 \rightarrow C_2$.

In parallel, $\mathcal{SDB}$ have been introduced. To support such a database, several architectures have been

proposed (Broekstra et al., 2002; Das et al., 2004; Jean et al., 2007; Pan and Heflin, 2003). They have been mainly focused on the scalability of these databases. Considering the support of ontology, each $\mathcal{SDB}$ supports the semantics of a given ontology model using hard-coded techniques. Therefore, the $\mathcal{SDB}$ may contain anomalies like redundant and inconsistent data. Unfortunately, there is no available methodology dedicated to the $\mathcal{SDB}$ design to reduce such anomalies.

Given account that the ontological concepts can be defined from each other, a dependency relationship between them may be synthesized. Two types of relations are identified: (i) the dependencies between ontological properties and (ii) the dependencies between ontological classes. By studying ontologies, we note that these models may include a set of concepts definitions that can be useful to identify dependencies relationships either between classes or properties. Some properties functional dependencies can not be synthesized based on these definitions. They can not be handled with the existing ontology models. To offer designers the means to express such dependencies and to define explicitly conceptual dependencies, we propose to extend the expressive power of ontologies by incorporating these new concepts in the ontology models and exploit them to design consistent $\mathcal{SDB}$ logical models. Such a design methodology is proposed in the next section.

3 DESIGN OF CONSISTENT SEMANTIC DATABASE LOGICAL MODELS

In this paper, we are interested in exploiting dependency relationships between ontological concepts to improve the semantic databases design process. Inspired from relational database design (Chen, 1975), we propose to incorporate dependency relationships in the $\mathcal{SDB}$ design process in order to (1) reduce redundancy by generating a normalized logical model, and (2) improve the database quality by detecting the inconsistent data caused by the violation of integrity constraints identified from dependencies definitions.

Considering the duplicated data stored in the $\mathcal{SDB}$, we deduce that redundancy is present due to (a) storing both data provided from canonical (CC) and non-canonical classes (NCC) and (b) the fixed logical model. Since several redundancy cases may be raised in the existing $\mathcal{SDB}$, we propose to proceed as follows.

1. First, the designer defines the conceptual model by extracting a fragment of the used ontology according to her/his requirements.
2. Second, we propose to exploit class dependencies to identify canonical an non-canonical classes. A dependency graph is firstly built from FD(C) where classes are nodes and class dependencies relationships describe the edges. This graph is then used as the input of our proposed algorithm to determine the minimum set of CC and generate the NCC. This algorithm starts by computing the isolated classes (classes not involved in FD(C)). Note that these classes will be canonical since they can not be derived from other ones. Then, the minimum coverage-like classes is computed. It represents the minimum subset of basic FD(C) to generate all the others. Finally, CC are identified and NCC are generated.
3. Third, for each canonical class $cc_i \in$ CC, we exploit the FD(R) defined on their properties to generate the normalized logical model. Note that, for each cc_i, a primary key is computed and relations in the 3NF are generated.
4. Fourth, to facilitate the user access, we define a view on the normalized relations corresponding to each canonical class. So, the users may query the $\mathcal{SDB}$ without worrying about the physical implementation of those classes.
5. Then, for each $ncc_i \in$ NCC, a relational view is computed. One of the advantages of using views to represent NCC is to ensure the transparency in accessing data and reduce redundancy in the target $\mathcal{SDB}$.
6. Finally, the quality of the stored data is studied. In fact, we observe that the inconsistent data may result from the violation of the FD. Based on the FD(R) defined for each ontological class, we propose to define rules taking into account the FD between properties in the ontological data storage. For each $FD(R)_i \in$ FD(R), we propose to define a rule avoiding the violation of the integrity constraint expressed by the $FD(R)_i$. This rule allows the designer to detect the inconsistent data by mentioning the constraint violation with displaying a comment. The Figure 1 describes the proposed approach. A case study of the deployment of our approach in a particular $\mathcal{SDB}$ is described in the next section.

4 A CASE STUDY

In this section, we propose a case study implementing our approach in Oracle 11g. The choice of this

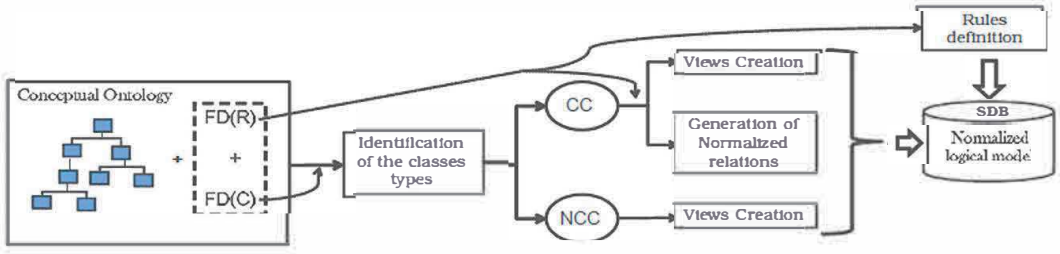

Figure 1: Semantic database design approach.

database is justified by the leading position that has Oracle 11g in the database area. During the validation process, we firstly present the different steps taken in the SDB design. Then, we present the contribution of our approach in such a database environment.

4.1 Design Steps

To validate our approach in Oracle 11g, a set of steps is required as follows: (1) extension of the OWL meta-schema, (2) primary key computation, (3) classes analysis, (4) conversion of the OWL ontology to a N-Triple format, (5) preparation stage, (6) ontology loading and (7) rules definition.

- **The OWL Meta-schema Extension.** Since the used ontology language does not handle dependencies representation, we propose to extend the OWL meta-schema by adding the meta-classes describing respectively a functional dependency defined between properties (FD(R)), its right part, its left part, a class dependency (FD(C)), the FD(C) right part and the FD(C) left part. For each added meta-class, a set of meta-properties is defined. These dependencies may be exploited to compute the types and the primary keys of classes. So, to represent such a data, we propose to enrich the OWL meta-model by adding the meta-classes PrimaryKey, NonCanonic and Canonic describing respectively the primary key concept and the canonicity of classes.
- **Compute Primary Keys.** To compute the appropriate primary key for each ontological class, we exploit the FD(R) defined on the properties of each class. To do so, we apply a java program that we have developed, on the extended OWL ontology. Based on the FD(R), this program computes and associates a primary key for each ontological class. To lead our validation, we use an extended fragment of the Lehigh University ontology. Let us assume that the *idUniv* property that describes the university identifier is generated as the primary key of the *University* class. Therefore, this step triggers the meta-schema instantiation by adding the appropriate ontological data describing the primary key definition. Once the primary keys are computed, an analysis to identify canonical and non-canonical classes is performed as described in the next step.
- **Analysis of Classes.** To specify the classes types, we apply a java program, exploiting the class dependencies, on the the Lehigh University ontology. Once the canonical and non-canonical classes are identified, we instantiate the meta-scheme classes "*Canonic*" and "*NonCanonic*" by generating the owl statements describing the classes types definition.
- **Ontology Conversion.** Oracle 11g offers only the data loading under the N-TRIPLE format (.nt). To meet this requirement, we use the converter *rdfcat* provided by the Jena API (version 2.6.4). This tool enables the transformation of an OWL file (.owl) to a N-TRIPLE file (.nt).
- **Preparation Stage.** Before the ontology loading, Oracle 11g requires a preparation step. This needs the sequence of a set of stages as follow: (1) activating the semantic module, (2) creating the semantic module, (3) creating the semantic table and (4) finally creating the semantic model.
- **Loading.** Oracle 11g offers several techniques for the ontology loading. We chose the bulk load for its fast loading. It loads the ontological data in a staging table using the SQLLoader utility (sqlldr) before being sent to the database.
- **Rules Definition.** Once the ontology loading is done, we propose the definition of a set of rules exploiting the property FD and defined primary keys to detect a set of inconsistent data violating these integrity constraints. These rules are defined in a base known as a rulebase in Oracle that raise the constraint violations defined through its rules application. For example, let us assume the existence of triples describing that University#1 and University#3 are two universities whose identifier values are both M50421 ((University#1, idUniv, M50421), (University#3, idUniv, M50421)). Note that their name values are respectively PoitiersUniversity and Tour-

sUniversity (University#1, name, PoitiersUniversity) and (University#3, name, ToursUniversity). Given that the *idUniv* property is computed as the primary key of the University class, the loaded triplets present a violation case of the uniqueness constraint. Indeed, the two universities are different but they have the same identifier. In order to detect such data, we propose the rule definition to address the primary key violation for each stored ontological class. The definition of such a rule requires the creation of a rulebase, followed by an index creation. This rule raises the uniquenessconstraint violation case by generating a comment to prevent users about the data inconsistency existence as follows: (s, :comment, 'Inconsistency: violation of primary key constraint') where s presents the inconsistent instance. Once the inconsistent data error is raised the user will decide if the data should be deleted or not.

4.2 Synthesis

In the most $\mathcal{SDB}$, storage models are frozen. For example, in Oracle 11g, the vertical representation is used for the storage of ontological concepts and instances referencing them. Therefore, our approach cann not be applied globally. This does not eliminate the contributions offered by our methodology for such databases. Indeed, based on the class dependencies, the class's types are identified and stored in the ontology-based data models. The property functional dependencies modeling allows to compute primary keys for each ontological class and subsequently, to store them in the $\mathcal{SDB}$. Based on the basis of these keys, the rules helping to detect a set of inconsistent data may be defined. They ensure that the stored data correspond to the boundaries of the modeled universe and reduce the inconsistency and redundancy in the $\mathcal{SDB}$ models.

5 CONCLUSIONS

This paper presents a complete methodology for designing $\mathcal{SDB}$ covering the three main steps of traditional database design: conceptual, logical and physical. Ontology is the core of our design methodology. Traditional definition of ontology is enriched by dependencies between properties and classes. These dependencies are exploited to generate consistent $\mathcal{SDB}$. They are used to identify the canonical concepts that have to be stored in the database. The non-canonical concepts are managed by relational views. A case study showing the deployment of our $\mathcal{SDB}$ obtained by our methodology in the semantic Oracle DBMS is proposed.

Currently, we are developing a design tool to assist designers during the $\mathcal{SDB}$ design process. Also, we are working on incorporating optimization structure selection during the physical design.

REFERENCES

Borgida, A. and Weddell, G. E. (1997). Adding uniqueness constraints to description logics (preliminary report). In *Deductive and Object-Oriented Databases, 5th International Conference (DOOD'97)*, pages 85–102.

Broekstra, J., Kampman, A., and Harmelen, F. V. (2002). Sesame: A generic architecture for storing and querying rdf and rdf schema. In *International Semantic Web Conference*, pages 54–68.

Calbimonte, J. P., Porto,F., and Keet, C. M. (2009). Functional dependencies in owl abox. In *Brazilian Symposium on Databases (SBBD)*, pages 16–30.

Calvanese, D., Giacomo, G. D., Lembo, D., Lenzerini, M., and Rosati, R. (2008). Path-based identification constraints in description logics. pages 231–241.

Chen, P. P. (1975). The enity-relationship model: Toward a unified view of data. In *VLDB*, page 173.

Das, S., Chong, E. I., Eadon, G., and Srinivasan, J. (2004). Supporting ontology-based semantic matching in rdbms. In *VLDB*, pages 1054–1065.

Dehainsala, H., Pierra, G., and Bellatreche, L. (2007). Ontodb: An ontology-based database for data intensive applications. In *DASFAA*, pages 497–508.

Jean, S., Dehainsala, H., Nguyen Xuan, D., Pierra, G., Bellatreche, L., and Aït-Ameur, Y. (2007). Ontodb: It is time to embed your domain ontology in your database. In *Proceedings of the 12th International Conference on Database Systems for Advanced Applications*, pages 1119–1120.

Motik, B., Horrocks, I., and Sattler, U. (2009). Bridging the gap between owl and relational databases. *Journal of Web Semantics*, 7(2):74–89.

Pan, Z. and Heflin, J. (2003). Dldb: Extending relational databases to support semantic web queries. In *The First International Workshop on Practical and Scalable Semantic Systems*.

Romero, O., Calvanese, D., Abelló, A., and Rodriguez-Muro, M. (2009). Discovering functional dependencies for multidimensional design. In *DOLAP*, pages 1–8.